

David Yevick

A First Course in Computational Physics and Object-Oriented Programming with C++

计算物理和C++面向对象的程序设计教程



CAMBRIDGE

世界图书出版公司

www.wpcbj.com.cn

A First Course in Computational Physics and Object-Oriented Programming with C++

It is the only C++ book that covers the entire range of topics



Cambridge
UNIVERSITY PRESS

978 0 521 87622 3
www.cambridge.org/9780521876223

A First Course in Computational Physics and Object-Oriented Programming with C++

David Yevick



图书在版编目 (CIP) 数据

计算物理和 C++ 面向对象的程序设计教程 = A first course in computational physics and objectoriented programming with C++ : 英文/ (美)耶维克 (Yevick, D.) 著. —北京: 世界图书出版公司北京公司, 2014. 10
ISBN 978 - 7 - 5100 - 8773 - 8

I. ①计… II. ①耶… III. ①C 语言—程序设计—教材—英文 IV. ① 04

中国版本图书馆 CIP 数据核字 (2014) 第 243085 号

书 名: A First Course in Computational Physics and Object-Oriented Programming with C++

作 者: David Yevick

中译名: 计算物理和 C++ 面向对象的程序设计教程

责任编辑: 高蓉 刘慧

出 版 者: 世界图书出版公司北京公司

印 刷 者: 三河市国英印务有限公司

发 行: 世界图书出版公司北京公司 (北京朝内大街 137 号 100010)

联系电话: 010 - 64021602, 010 - 64015659

电子信箱: kjb@wpcbj.com.cn

开 本: 16 开

印 张: 27

版 次: 2015 年 3 月

版权登记: 图字: 01 - 2013 - 4909

书 号: 978 - 7 - 5100 - 8773 - 8

定 价: 99.00 元

A First Course in Computational Physics and Object-Oriented Programming with C++

Because of its rich object-oriented features, C++ is rapidly becoming the programming language of choice for science and engineering applications. This text leads beginning and intermediate programmers step by step through the difficult aspects of scientific coding, providing a comprehensive survey of object-oriented methods.

Numerous aspects of modern programming practice are covered, including object-oriented analysis and design tools, numerical analysis, scientific graphics, software engineering, performance issues, and legacy software reuse. Examples and problems are drawn from an extensive range of scientific and engineering applications. An emphasis on the fundamental logical principles of the language, combined with short, focused illustrations and discussions, helps promote rapid learning. The book also includes a CD-ROM with a full set of free programming and scientific graphics tools that facilitate individual learning and reduce the time required to supervise code development in a classroom setting.

This unique text will be invaluable both to students taking a first or second course in computational science and as a reference text for scientific programmers.

DAVID YEVIK is a leading scientist in the application of numerical modeling to optical communication systems, notably guided electric field propagation in waveguides and fibers, optical processes in semiconductors, and most recently optical communication systems. Over the last 25 years, he has collaborated with numerous industrial and government research establishments, including spending four years with the IBM Center of Advanced Studies, Canada, on practical applications of the VisualAge for C++ toolset.

Dr. Yevick has extensive teaching experience in both electrical engineering and physics. Since 1999, he has been a full professor of physics at the University of Waterloo, Ontario having previously held positions in the Electrical and Computer Engineering Departments of Queen's University at Kingston and the Pennsylvania State University, in the Solid State Physics Department of Lunds Universitet and at the Institutet för Optisk Forskning, Stockholm. He has authored or co-authored over 120 refereed journal articles. Dr. Yevick is a fellow of the American Physical Society, the Institute of Electrical and Electronics Engineers, and the Optical Society of America and is a registered Professional Engineer, Ontario.

A First Course in Computational Physics and Object-Oriented Programming with C ++ (978-0-521-82778-2) by David Yevick, first published by Cambridge University Press 2005

All rights reserved.

This reprint edition for the People's Republic of China is published by arrangement with the Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.

© Cambridge University Press & Beijing World Publishing Corporation 2014

This book is in copyright. No reproduction of any part may take place without the written permission of Cambridge University Press or Beijing World Publishing Corporation.

This edition is for sale in the mainland of China only, excluding Hong Kong SAR, Macao SAR and Taiwan, and may not be bought for export therefrom.

此版本仅限中华人民共和国境内销售，不包括香港、澳门特别行政区及中国台湾。不得出口。

To my parents, George and Miriam, who were the first to teach me physics and mathematics, my wife Susan for her unending support and to Ariela, Hannah and Aaron who I hope one day will find this book useful

Contents

| | |
|---|---------------|
| Part I C++ programming basics | <i>page 1</i> |
| 1 Introduction | 3 |
| 1.1 Objective | 3 |
| 1.2 Presentation | 3 |
| 1.3 Why C++ | 4 |
| 1.4 C++ standards | 6 |
| 1.5 Summary | 7 |
| 1.6 How to use this text | 7 |
| 1.7 Additional study aids | 8 |
| 1.8 Additional and alternative software packages | 8 |
| 2 Installing and running the Dev-C++ programming environment | 10 |
| 2.1 Compiling and running a first program | 10 |
| 2.2 Using the Dev-C++ debugger | 12 |
| 2.3 Installing DISLIN and gsl | 13 |
| 2.4 A first graphics program | 14 |
| 2.5 The help system | 15 |
| 2.6 Linux alternatives | 16 |
| 2.7 Assignment | 16 |
| 3 Introduction to computer and software architecture | 17 |
| 3.1 Computational methods | 17 |
| 3.2 Hardware architecture | 18 |
| 3.3 Software architecture | 20 |
| 3.4 The operating system and application software | 23 |
| 3.5 Assignments | 23 |
| 4 Fundamental concepts | 25 |
| 4.1 Overview of program structure | 25 |
| 4.2 Tokens, names, and keywords | 25 |
| 4.3 Expressions and statements | 26 |
| 4.4 Constants, variables, and identifiers | 26 |
| 4.5 Declarations, definitions, and scope | 27 |

| | | |
|----------|--|-----------|
| 4.6 | rvalues and lvalues | 28 |
| 4.7 | Block structure | 28 |
| 4.8 | The const keyword | 29 |
| 4.9 | Operators – precedence and associativity | 30 |
| 4.10 | Formatting conventions | 31 |
| 4.11 | Comments | 32 |
| 4.12 | Assignments | 33 |
| 5 | Writing a first program | 37 |
| 5.1 | The main() function | 37 |
| 5.2 | Namespaces | 37 |
| 5.3 | #include Statements | 38 |
| 5.4 | Input and output streams | 39 |
| 5.5 | File streams | 40 |
| 5.6 | Constant and variable types | 41 |
| 5.7 | Casts | 44 |
| 5.8 | Operators | 45 |
| 5.9 | Control flow | 46 |
| 5.10 | Functions | 47 |
| 5.11 | Arrays and typedefs | 47 |
| 5.12 | A first look at scientific software development | 48 |
| 5.13 | Program errors | 51 |
| 5.14 | Numerical errors with floating-point types | 53 |
| 5.15 | Assignments | 55 |
| 6 | An introduction to object-oriented analysis | 62 |
| 6.1 | Procedural versus object-oriented programming | 62 |
| 6.2 | Problem definition | 65 |
| 6.3 | Requirements specification | 66 |
| 6.4 | UML diagrams | 66 |
| 6.5 | Use case diagram | 67 |
| 6.6 | Classes and objects | 68 |
| 6.7 | Object discovery | 71 |
| 6.8 | Sequence and collaboration diagrams | 72 |
| 6.9 | Aggregation and association | 74 |
| 6.10 | Inheritance | 75 |
| 6.11 | Object-oriented programming approaches | 78 |
| 6.12 | Assignments | 79 |
| 7 | C++ object-oriented programming syntax | 83 |
| 7.1 | Class declaration | 83 |
| 7.2 | Class definition and member functions | 83 |
| 7.3 | Object creation and polymorphism | 86 |

| | | |
|-----------|---|------------|
| 7.4 | Information hiding | 87 |
| 7.5 | Constructors | 89 |
| 7.6 | Wrapping legacy code | 91 |
| 7.7 | Inheritance | 92 |
| 7.8 | The 'protected' keyword | 94 |
| 7.9 | Assignments | 95 |
| 8 | Control logic and iteration | 104 |
| 8.1 | The bool and enum types | 104 |
| 8.2 | Logical operators | 106 |
| 8.3 | if statements and implicit blocks | 107 |
| 8.4 | else , else if , conditional and switch statements | 108 |
| 8.5 | The exit() function | 109 |
| 8.6 | Conditional compilation | 109 |
| 8.7 | The for statement | 110 |
| 8.8 | while and do...while statements | 111 |
| 8.9 | The break and continue statements | 112 |
| 8.10 | Assignments | 112 |
| 9 | Basic function properties | 119 |
| 9.1 | Principles of function operation | 119 |
| 9.2 | Function declarations and prototypes | 121 |
| 9.3 | Overloading and argument conversion | 121 |
| 9.4 | Built-in functions and header files | 122 |
| 9.5 | Program libraries | 124 |
| 9.6 | Function preconditions and postconditions – the assert statement | 128 |
| 9.7 | Multiple return statements | 130 |
| 9.8 | Functions and global variables | 130 |
| 9.9 | Use of const in functions | 131 |
| 9.10 | Default parameters | 132 |
| 9.11 | Inline functions | 132 |
| 9.12 | Modular programming | 133 |
| 9.13 | Recursive functions | 134 |
| 9.14 | Assignments | 134 |
| 10 | Arrays and matrices | 140 |
| 10.1 | Data structures and arrays | 140 |
| 10.2 | Array definition and initialization | 141 |
| 10.3 | Array manipulation and memory access | 142 |
| 10.4 | Arrays as function parameters | 144 |
| 10.5 | Returning arrays and object arrays | 145 |
| 10.6 | const arrays | 146 |
| 10.7 | Matrices | 146 |

| | | |
|----------------|---|------------|
| 10.8 | Matrix storage and loop order | 147 |
| 10.9 | Matrices as function arguments | 150 |
| 10.10 | Assignments | 150 |
| 11 | Input and output streams | 158 |
| 11.1 | The <code>iostream</code> class and stream manipulators | 158 |
| 11.2 | File streams | 161 |
| 11.3 | The <code>string</code> class and string streams | 163 |
| 11.4 | The <code>toString()</code> class member | 165 |
| 11.5 | The <code>printf</code> function | 167 |
| 11.6 | Assignments | 167 |
| Part II | Numerical analysis | 173 |
| 12 | Numerical error analysis – derivatives | 175 |
| 12.1 | The derivative operator | 175 |
| 12.2 | Error dependence | 177 |
| 12.3 | Graphical error analysis | 177 |
| 12.4 | Analytic error analysis – higher-order methods | 179 |
| 12.5 | Extrapolation | 179 |
| 12.6 | The derivative calculator class | 180 |
| 12.7 | Assignments | 182 |
| 13 | Integration | 183 |
| 13.1 | Discretization procedures | 183 |
| 13.2 | Implementation | 184 |
| 13.3 | Discretization error | 188 |
| 13.4 | Assignments | 189 |
| 14 | Root-finding procedures | 191 |
| 14.1 | Bisection method | 191 |
| 14.2 | Newton’s method | 193 |
| 14.3 | Assignments | 194 |
| 15 | Differential equations | 196 |
| 15.1 | Euler’s method | 196 |
| 15.2 | Error analysis | 198 |
| 15.3 | The spring class | 199 |
| 15.4 | Assignments | 201 |
| 16 | Linear algebra | 203 |
| 16.1 | Linear equation solvers | 203 |

| | | |
|--|--|------------|
| 16.2 | Errors and condition numbers | 206 |
| 16.3 | Eigenvalues and iterative eigenvalue solvers | 207 |
| 16.4 | Assignments | 209 |
| Part III Advanced object-oriented programming | | 215 |
| 17 | References | 217 |
| 17.1 | Basic properties | 217 |
| 17.2 | References as function arguments | 218 |
| 17.3 | Reference member variables | 219 |
| 17.4 | const reference variables | 220 |
| 17.5 | Reference return values | 221 |
| 17.6 | Assignments | 222 |
| 18 | Pointers and dynamic memory allocation | 227 |
| 18.1 | Introduction to pointers | 228 |
| 18.2 | Initializing pointer variables | 228 |
| 18.3 | The address-of and dereferencing operators | 229 |
| 18.4 | Uninitialized pointer errors | 230 |
| 18.5 | NULL and void pointers | 230 |
| 18.6 | Dangling pointers | 231 |
| 18.7 | Pointers in function blocks | 232 |
| 18.8 | The const keyword and pointers | 232 |
| 18.9 | Pointer arithmetic | 234 |
| 18.10 | Pointers and arrays | 234 |
| 18.11 | Pointer comparisons | 234 |
| 18.12 | Pointers to pointers and matrices | 235 |
| 18.13 | String manipulation | 235 |
| 18.14 | Static and dynamic memory allocation | 237 |
| 18.15 | Memory leaks and dangling pointers | 239 |
| 18.16 | Dynamic memory allocation within functions | 241 |
| 18.17 | Dynamically allocated matrices | 242 |
| 18.18 | Dynamically allocated matrices as function arguments and parameters | 243 |
| 18.19 | Pointer data structures and linked lists | 244 |
| 18.20 | Assignments | 248 |
| 19 | Advanced memory management | 261 |
| 19.1 | The this pointer | 261 |
| 19.2 | The friend keyword | 262 |
| 19.3 | Operators | 263 |
| 19.4 | Destructors | 265 |

| | | |
|----------------|---|------------|
| 19.5 | Assignment operators | 267 |
| 19.6 | Copy constructors | 269 |
| 19.7 | Assignments | 271 |
| 20 | The static keyword, multiple and virtual inheritance, templates, and the STL library | 286 |
| 20.1 | Static variables | 286 |
| 20.2 | Static class members | 287 |
| 20.3 | Pointer to class members | 288 |
| 20.4 | Multiple inheritance | 288 |
| 20.5 | Virtual functions | 289 |
| 20.6 | Heterogeneous object collections and runtime type identification | 291 |
| 20.7 | Abstract base classes and interfaces | 292 |
| 20.8 | Virtual inheritance | 293 |
| 20.9 | User-defined conversions | 294 |
| 20.10 | Function templates | 295 |
| 20.11 | Templates and classes | 296 |
| 20.12 | The complex class | 298 |
| 20.13 | The standard template library | 299 |
| 20.14 | Structures and unions | 303 |
| 20.15 | Bit fields and operators | 305 |
| 20.16 | Assignments | 306 |
| 21 | Program optimization in C++ | 319 |
| 21.1 | Compiling | 319 |
| 21.2 | Critical code segments | 319 |
| 21.3 | Virtual functions | 321 |
| 21.4 | Function pointers and functors | 326 |
| 21.5 | Aliasing | 327 |
| 21.6 | High-performance template libraries | 327 |
| 21.7 | Assignments | 329 |
| Part IV | Scientific programming examples | 331 |
| 22 | Monte Carlo methods | 333 |
| 22.1 | Monte Carlo integration | 333 |
| 22.2 | Monte Carlo evaluation of distribution functions | 334 |
| 22.3 | Importance sampling | 339 |
| 22.4 | The metropolis algorithm | 343 |
| 22.5 | Multicanonical methods | 347 |
| 22.6 | Assignments | 352 |

| | |
|--|-----|
| 23 Parabolic partial differential equation solvers | 354 |
| 23.1 Partial differential equations in scientific applications | 354 |
| 23.2 Direct solution methods | 356 |
| 23.3 The Crank–Nicholson method | 359 |
| 23.4 Assignments | 363 |
| Appendix A Overview of MATLAB | 365 |
| Appendix B The Borland C++ Compiler | 371 |
| B.1 Borland C++ installation | 371 |
| B.2 Compiling and running a first program | 373 |
| B.3 Installing the optional program editor | 375 |
| B.4 Using the Borland turbo debugger | 377 |
| B.5 Installing DISLIN | 378 |
| B.6 A first graphics program | 379 |
| B.7 The help system | 380 |
| Appendix C The Linux/Windows Command-Line C++ Compiler and Profiler | 381 |
| Appendix D Calling FORTRAN programs from C++ | 384 |
| Appendix E C++ coding standard | 387 |
| E.1 Program design language | 387 |
| E.2 Comments | 388 |
| E.3 Layout | 388 |
| E.4 Continuation lines | 389 |
| E.5 Constants and literals | 389 |
| E.6 Variables and definitions | 389 |
| E.7 Functions | 390 |
| E.8 Operators | 390 |
| E.9 Classes | 390 |
| E.10 Typedefs | 391 |
| E.11 Macros | 391 |
| E.12 Templates | 391 |
| E.13 Control structures | 391 |
| <i>References and further reading</i> | 393 |
| <i>Index</i> | 398 |

Part I

C++ programming basics

Chapter 1

Introduction

This textbook is the result of seven years of experience with teaching scientific programming in both science and engineering departments. The book has a single, clearly defined goal – to convey as broad an understanding of the entire scientific computing field as possible within a single term course. Accordingly, while the C++ programming language is explained concisely, its conceptual foundation is emphasized. Once this framework is understood, the complex language syntax can be far more easily retained. Further, all features of modern programming of relevance to scientific programming are surveyed with emphasis on strategies for simplifying coding. Free software tools are included that minimize the technical hurdles of coding and running programs.

1.1 Objective

As stated above, this textbook presents a broad introduction to modern scientific programming. This includes numerical analysis, object-oriented programming, scientific graphics, software engineering, and the modeling of advanced physical systems. Consequently, knowledge of the material will provide sufficient background to enable the reader to analyze and solve nearly all normally encountered scientific programming tasks.

1.2 Presentation

This text is concise, focusing on essential concepts. Examples are intentionally short and free of extraneous features. To promote retention, the book repeats key topics in cycles of gradually increasing difficulty. Further, since the process of learning computer language shares many similarities with that of acquiring a spoken language, the most important sample program segments are highlighted in gray. *Memorizing these greatly decreases the time required to achieve proficiency in C++ programming.*